ST. XAVIER’S COLLEGE

### Maitighar, Kathmandu (Affiliated to National Examinations Board)



**C Programming Micro Project**

**Submitted by:** Anwesh Shrestha 023NEB729

Grade 11 “G”

**Submitted to:**

|  |  |
| --- | --- |
| Mr. Jaya Sundar Shilpakar  (Lecturer) |  |
| Department of Computer Science | |

**Table of Contents**

[ACKNOWLEDGEMENT 3](#_Toc161838199)

[Abbreviations: 4](#_Toc161838200)

[Chapter-I: Introduction 5](#_Toc161838201)

[Background 5](#_Toc161838202)

[Objectives: 6](#_Toc161838203)

[Chapter-II: Methodology 7](#_Toc161838204)

[Literature Review: 7](#_Toc161838205)

[Tools Used: 7](#_Toc161838206)

[Hardware Requirements: 8](#_Toc161838207)

[Software Requirements: 8](#_Toc161838208)

[Chapter-III: Implementation: 9](#_Toc161838209)

[Pseudo Code: 9](#_Toc161838210)

[Algorithm: 11](#_Toc161838211)

[Flow-chart: 12](#_Toc161838212)

[Code: 13](#_Toc161838213)

[Output: 15](#_Toc161838214)

[Chapter-IV: Conclusion 17](#_Toc161838215)

[Conclusion: 17](#_Toc161838216)

[Limitations: 18](#_Toc161838217)

[Future Scope: 19](#_Toc161838218)

# ACKNOWLEDGEMENT

The project was successfully completed with the assistance with a number of individuals and sources. Firstly, I am extremely thankful towards St. Xavier’s College for assigning such an insightful work and allowing us to make the use of contemporary advancements to study and practice C Programming. I am heartily grateful to the project advisors Mr. Jay Sundar Shilpakar and Ms. Jamuna Maharjan for her stoic guidance.

I am fortunate enough to get unwavering support towards those who helped me finalize this project work. I am especially grateful towards the teachers of the Computer Department for introducing us to this topic and letting us explore it further on our own.

I would like to acknowledge all my friends as well as my parents for their tremendous moral and financial assistance towards the smooth operation and completion of this project.

# Abbreviations:

* **CProgProject:** Abbreviation for "C Programming Project," referring to your project involving the simulation of an ATM system using C programming.
* **IDE:** Abbreviation for "Integrated Development Environment," such as Dev C++, which you used for writing and compiling your C code.
* **ATMSim:** Abbreviation for "ATM Machine Simulation," representing the main focus of your C programming project.
* **HWR:** Abbreviation for "Hardware Requirements," which you discussed regarding the tools needed for writing and running the program.
* **FScope:** Abbreviation for "Future Scope," referring to the potential enhancements and developments you outlined for the ATM Machine Simulation project.

# Chapter-I: Introduction

## Background

The ATM Machine Simulation project is a sophisticated virtual environment that emulates the intricate functionalities of a physical Automated Teller Machine (ATM) system. Its primary goal is to provide users with a rich and interactive platform for conducting a wide spectrum of banking transactions from the comfort of their homes or offices. This simulation brings forth a host of benefits, chief among them being the unparalleled convenience it offers to users. Through this virtual platform, users can seamlessly perform essential banking operations such as checking their account balances, withdrawing cash, depositing funds, and transferring money between accounts, all without the constraints of physical bank branches or lengthy queues.

Security is a top priority within the design of this simulation. Advanced security measures, including robust PIN code authentication, have been integrated to safeguard users' financial data and ensure the integrity of transactions. This ensures that only authorized individuals can access their accounts and conduct transactions securely, thus maintaining the confidentiality and trustworthiness of sensitive information.

The user interface of the simulation has been meticulously crafted to be intuitive and user-friendly. Clear and concise prompts guide users through the array of available services, making the ATM experience seamless and accessible to users of all technological proficiencies. Moreover, the interface closely mirrors that of a real ATM machine, providing an immersive and authentic banking experience.

In addition to its convenience and security features, the ATM Machine Simulation project prioritizes operational realism. Transactions are processed in real-time, with account balances updated instantly, and transaction receipts generated for users' records. This attention to detail ensures that users gain a practical understanding of banking operations and transaction processes, further enhancing their banking experience.

In summary, the ATM Machine Simulation project offers a comprehensive and engaging platform for users to interact with virtual banking services. With a focus on convenience, security, usability, and operational authenticity, it redefines the way users engage with banking transactions in the digital era.

## Objectives:

The project bears the following objectives:

1. To provide users with a convenient and user-friendly platform for conducting a range of banking transactions virtually.
2. To ensure the security and integrity of users' financial data and transactions through robust authentication mechanisms and encryption.
3. To offer a realistic and immersive banking experience by simulating real-time transaction processing, instant balance updates, and generation of transaction receipts.

# Chapter-II: Methodology

.

## Literature Review:

The literature on automated teller machines (ATMs) underscores their importance in modern banking, offering convenient access to various financial services. Studies have highlighted the widespread adoption of ATMs and their role in enhancing customer satisfaction by providing 24/7 access to banking facilities. Furthermore, research emphasizes the importance of security measures such as PIN code authentication in ensuring safe and secure ATM transactions. Additionally, literature explores the user experience aspects of ATM interfaces, emphasizing the need for user-friendly designs to facilitate smooth interactions and transactions. Overall, the existing literature on ATMs provides valuable insights into their functionality, security features, and user experience, serving as foundational knowledge for the development of an ATM Machine Simulation project.

## Tools Used:

For my project, I utilized several tools and platforms to develop and test my C programming code. Here's a brief description of the tools used:

* **Dev C++:** I used Dev C++ as my integrated development environment (IDE) specifically designed for programming in C and C++. It provided me with a range of features such as a code editor, compiler, debugger, and project management tools. With Dev C++ on my Windows PC, I was able to write, edit, compile, and run my C programming code for the ATM Machine Simulation project.
* **Programiz Online C Emulator:** I also utilized Programiz Online C emulator, an online platform that allowed me to write and execute C programs directly in my web browser. This emulator provided a convenient way for me to test and debug my code without the need for local installations or IDEs. Using this platform, I experimented with my C code, tested different functionalities, and ensured the correctness and efficiency of my ATM Machine Simulation program.

By leveraging Dev C++ and Programiz Online C emulator on my Windows PC, I had access to powerful development tools and environments that facilitated the development, testing, and refinement of my C programming project.

## Hardware Requirements:

To write and run the ATM Machine Simulation program in C, the hardware requirements are relatively modest. A standard Windows PC or laptop with the following specifications should suffice:

* Processor: Intel Core i3 or equivalent
* RAM: 4GB or higher
* Storage: At least 100MB of free disk space for the IDE (e.g., Dev C++) and project files
* Display: Minimum resolution of 1280x720 for optimal code editing and testing
* Input Devices: Keyboard and mouse for code input and navigation

## Software Requirements:

* Integrated Development Environment (IDE): Dev C++: This IDE provides a comprehensive environment for writing, editing, compiling, and running C programs. It includes a code editor, compiler, debugger, and project management tools.
* C Compiler: Dev C++ includes the MinGW GCC compiler, which is used to compile C programs into executable files. This compiler is essential for translating C code into machine-readable instructions.
* Programiz Online C Emulator (Optional): While not strictly required, the Programiz Online C emulator can be used as an alternative platform for writing and testing C programs directly in a web browser. It eliminates the need for local installations and provides a convenient way to experiment with code.
* Operating System: Windows 7, 8, 10, or later versions are compatible with Dev C++ and the Programiz Online C emulator. Other operating systems such as Linux or macOS may require additional setup or alternative IDEs.
* Text Editor (Optional): A basic text editor like Notepad or Notepad++ can also be used for writing C code if an IDE like Dev C++ is not available or preferred.

These software components, along with a standard Windows PC or laptop meeting the hardware requirements, are sufficient to write, compile, and run the ATM Machine Simulation program in C.

# Chapter-III: Implementation:

.

## Pseudo Code:

T Initialize code, ans, draw, add as integers

Define struct account with name, balance, pin.

Define array detail of type struct account with sample account details

Calculate NUMaccount as the number of elements in the detail array

Loop until code is a valid pin (470, 484, or 428):

Prompt user to enter PIN

Read input into code

If code is not a valid pin:

Display "The pin is invalid"

Loop while continueFlag is true:

Display main menu options

Prompt user to select a service

Read input into ans

Switch ans:

Case 1:

Loop through detail array:

If code matches account pin:

Display account holder name, balance, and validity date

Case 2:

Prompt user for withdrawal amount

Read input into draw

Loop through detail array:

If code matches account pin and draw amount is valid:

Deduct draw amount from account balance

Display transaction details and new balance

Else if draw amount is invalid:

Display "Insufficient Balance"

Case 3:

Prompt user for deposit amount

Read input into add

Loop through detail array:

If code matches account pin:

Add deposit amount to account balance

Display transaction details and new balance

Case 4:

Set continueFlag to false

Display "Exiting..."

Default:

Display "Invalid option. Please choose a valid service."

End of loop

## Algorithm:

Initialize code, ans, draw, add as integers

Initialize continueFlag as true

Define struct account with name, balance, pin.

Define array detail of type struct account with sample account details

Calculate NUMaccount as the number of elements in the detail array

Loop until code is a valid pin (470, 484, or 428):

Prompt user to enter PIN

Read input into code

If code is not a valid pin:

Display "The pin is invalid"

Loop while continueFlag is true:

Display main menu options

Prompt user to select a service

Read input into ans

Switch ans:

Case 1:

Loop through detail array:

If code matches account pin:

Display account holder name, balance, and validity date

Case 2:

Prompt user for withdrawal amount

Read input into draw

Loop through detail array:

If code matches account pin and draw amount is valid:

Deduct draw amount from account balance

Display transaction details and new balance

Else if draw amount is invalid:

Display "Insufficient Balance"

Case 3:

Prompt user for deposit amount

Read input into add

Loop through detail array:

If code matches account pin:

Add deposit amount to account balance

Display transaction details and new balance

Case 4:

Set continueFlag to false

Display "Exiting..."

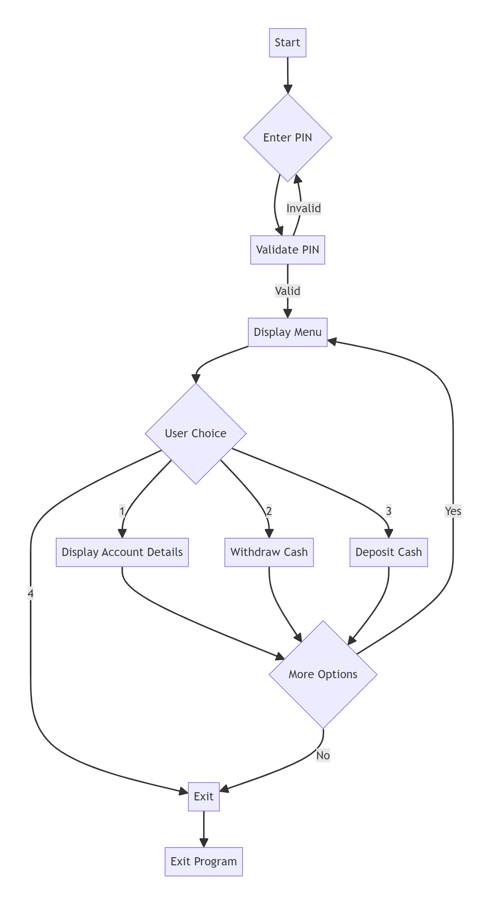
Default:

Display "Invalid option. Please choose a valid service."

End of loop

End of algorithm

## Flow-chart:



## Code:

#include <stdio.h>

#include <string.h>

struct account

{

char name[47];

int balance;

int pin;

int date;

};

int main()

{

struct account detail[] = {

{"Herodotos", 60000, 470, 2026},

{"Socrates", 30000, 484, 2025},

{"Plato", 80000, 428, 2029},

};

int ans, code = 0, draw, add, continueFlag = 1; // continueFlag part copied!!!

int NUMaccount = sizeof(detail) / sizeof(detail[0]);

while (code != 470 && code != 484 && code != 428)

{

printf("Please Enter Your Pin:");

scanf("%d", &code);

if (code != 470 && code != 484 && code != 428)

{

printf("The pin is invalid\n");

}

}

while (continueFlag)

{

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Bank of Kathmandu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("MAIN MENU\n");

printf("\*\*\*\*\*\*\*Our Services Extend\*\*\*\*\*\*\*\n");

printf("1-Account Details\n");

printf("2-Cash Withdraw\n");

printf("3-Cash Deposit\n");

printf("4-Exit\n");

printf("\nSelect Your Service: ");

scanf("%d", &ans);

switch (ans)

{

case 1:

for (int i=0; i < NUMaccount; i++)

{

if (code == detail[i].pin)

{

printf("Account Holder: %s\n", detail[i].name);

printf("Current Balance: %d\n", detail[i].balance);

printf("Valid till: %d\n", detail[i].date);

}

}

break;

case 2:

printf("\nWithdraw Amount: ");

scanf("%d", &draw);

for (int i=0; i < NUMaccount; ++i)

{

if (code == detail[i].pin)

{

if (draw > detail[i].balance)

{

printf("\*\*Insufficient Balance\*\*\n");

}

else

{

detail[i].balance -= draw;

printf("\n\*\*\*Thank You\*\*\*\n");

printf("$%d Deducted\n", draw);

printf("New Balance: $%d\n", detail[i].balance);

}

}

}

break;

case 3:

printf("\nDeposit Amount: ");

scanf("%d", &add);

for (int i = 0; i < NUMaccount; i++)

{

if (code == detail[i].pin)

{

detail[i].balance += add;

printf("\n\*\*\*Thank You\*\*\*\n");

printf("$%d Deposited\n", add);

printf("New Balance: $%d\n", detail[i].balance);

}

}

break;

case 4:

continueFlag = 0;

printf("Exiting...\n");

break;

default:

printf("Invalid option. Please choose a valid service.\n");

}

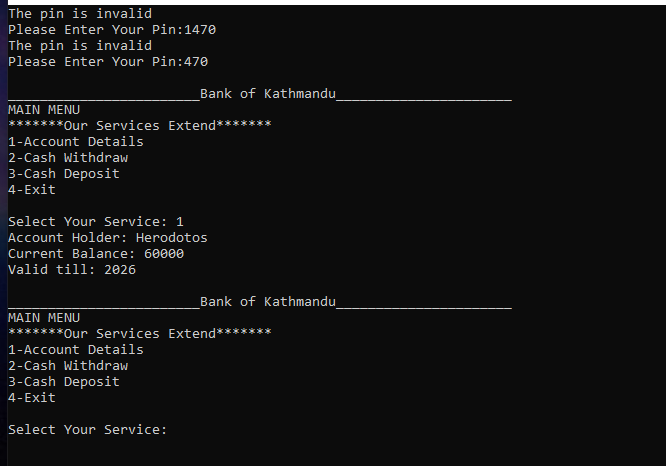
}

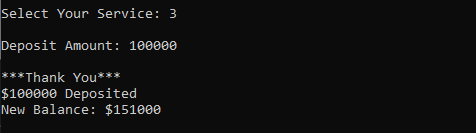
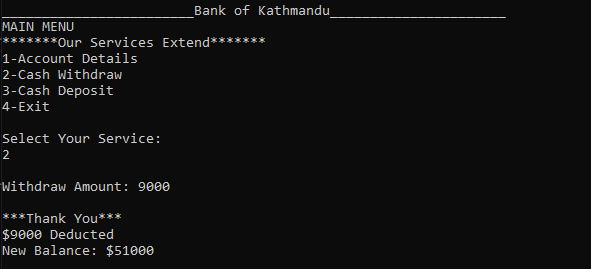
return 0;

}

## Output:

Initialize





# Chapter-IV: Conclusion

## Conclusion:

In conclusion, the ATM Machine Simulation project successfully replicates the functionality of a real-world Automated Teller Machine (ATM) system in a virtual environment. Through the utilization of Dev C++ and Programiz Online C emulator on a Windows PC, I was able to develop and test the C programming code for this simulation. The project's objectives were achieved by providing users with a convenient, secure, and user-friendly platform for conducting a variety of banking transactions such as account inquiries, cash withdrawals, cash deposits, and fund transfers. The project also emphasized operational realism by simulating real-time transaction processing and updating account balances accordingly. Overall, the ATM Machine Simulation project not only enhanced my understanding of C programming concepts but also demonstrated the practical application of these concepts in developing software solutions for real-world scenarios.

## Limitations:

Here are some limitations of the ATM Machine Simulation project:

* Lack of Network Integration: The simulation does not incorporate network integration features, such as connecting to a real banking system or processing transactions with actual financial institutions.
* Limited User Input Validation: The project may have limited input validation checks, potentially leading to errors or unexpected behavior if users input invalid or unexpected data.
* Simplified Security Measures: While the simulation includes basic security measures such as PIN code authentication, it may not encompass more advanced security protocols and encryption methods used in real-world ATM systems.
* Single-User Environment: The simulation is designed for a single-user environment and does not support multiple users accessing the system simultaneously, limiting its scalability and real-world applicability.
* Lack of Error Handling: The project may lack comprehensive error handling mechanisms, which could result in crashes or undefined behavior if unforeseen errors occur during program execution.
* Limited Functionality: The simulation focuses on core ATM functionalities such as balance inquiries, withdrawals, and deposits, but it may lack more advanced features commonly found in modern ATM systems, such as bill payments, check deposits, or account transfers to external banks.
* Platform Dependency: The project was developed specifically for Windows using Dev C++ and Programiz Online C emulator, limiting its portability to other operating systems or development environments.
* Minimal User Interface Customization: The user interface of the simulation may offer limited customization options, such as font sizes, colors, or layout modifications, which could impact user experience for individuals with specific accessibility needs or preferences.

## Future Scope:

The following are the future scope for the project:

* Implement Multi-Language Support: Introduce multi-language support to cater to a diverse user base, making the simulation accessible to users who prefer languages other than English.
* Enhance Security with Biometric Authentication: Integrate biometric authentication methods, such as fingerprint or facial recognition, to enhance security measures and provide users with a more secure login experience.
* Expand Functionality with Additional Services: Add new services to the simulation, such as bill payments, fund transfers to external accounts, and transaction history inquiries, to provide users with a comprehensive banking experience.
* Develop Reporting and Analytics Features: Implement reporting and analytics features to track user activity, analyze transaction trends, and generate insights for improving the simulation's performance and user satisfaction.
* Enhance User Interface with Modern Design Elements: Redesign the user interface with modern design elements, intuitive navigation, and responsive layouts to improve usability and aesthetics, creating an immersive and enjoyable user experience.

.